5/3-63
10P
188182

Task Planning and Control Synthesis for Robotic Manipulation in Space Applications

A.C. Sanderson, M.A. Peshkin, and L.S. Homem-de-Mello
Carnegie-Mellon University
Pittsburgh, PA 15213

CH 188052

1. Abstract

Space-based robotic systems for diagnosis, repair and assembly of systems will require new techniques of planning and manipulation to accomplish these complex tasks. ~~This paper summarizes~~ results of work in assembly task representation, discrete task planning, and control synthesis which provide a design environment for flexible assembly systems in manufacturing applications, and which extend to planning of manipulation operations in unstructured environments. Assembly planning is carried out using the AND/OR graph representation which encompasses all possible partial orders of operations and may be used to plan assembly sequences. Discrete task planning uses the *configuration map* which facilitates search over a space of discrete operations parameters in sequential operations in order to achieve required goals in the space of bounded configuration sets.

2. Introduction

Space-based robotic systems will be required to perform tasks involving dexterity, perception, and planning. Telerobotic systems integrate human perception and human planning capabilities in order to accomplish tasks. Autonomous systems will increasingly require imbedded task planning systems with accompanying sensory integration, control synthesis, and system architecture to support goal-directed activities in an uncertain environment. Diagnosis, repair, and assembly are tasks which will be essential to the maintenance of space-based systems and require both complex manipulation as well as reasoning about system configuration and system functionality. This paper reviews our recent work on task planning for assembly systems and discusses its implications for the development of robotic systems for assembly, maintenance, repair, and transport tasks.

Both manned and unmanned spacecraft require a variety of maintenance and repair tasks including materials handling, diagnosis of faults, reasoning about the origin of faults, hypothesis formation and testing, planning and executing repair procedures, disassembly, assembly, and replacement of parts. Currently these tasks may be accomplished in a limited way by on-site manual and teleoperated systems. As the number, complexity, cost, and importance of these spacecraft increases, autonomous systems which can provide service and maintenance on a routine basis will become essential.

Diagnosis and repair are problems in reasoning as well as manipulation. Any successful approach to these issues requires a representation of the task and an automated reasoning system which enables a decomposition of the problem into feasible sensing and manipulation procedures. Our work on assembly planning is based on several generations of assembly workcells which we built and demonstrated for manufacturing applications [1]. These flexible workcells incorporated multiple robot arms, vision, tactile and force sensing to accomplish tasks in electronic assembly, wire harness assembly, and assembly of instrument products such as copiers and printers.

Our experience with implementing tasks on these prototype workcells is the basis for current research on the development of tools for efficient design, programming, and implementation of complex systems. Task representation, decomposition, and sequencing [2,3,4], discrete task planning, [8] and adaptive control and learning techniques [9] are principal issues which are currently being addressed. Embedding such adaptation and learning procedures in the control and planning hierarchy is fundamental to successful implementation in uncertain environments. In this paper, we summarize an approach to assembly task representation and sequencing, and describe in more detail the use of the configuration map as a tool in discrete task planning.

The control functions of the system are allocated hierarchically into Strategic, Tactical, Operational, and Device levels. The control synthesis problem is to map the control hierarchy onto the set of feasible assembly plans in order to achieve desired performance. In this procedure, we seek to iteratively adjust the assignment of system resources subject to task precedence and configuration tolerance contraints. This procedure requires the definition of motion strategies and motion primitives which can be employed. We have developed a detailed understanding of sensorless manipulation strategies [5,6,7,8] which facilitate planning of sliding, pushing, and grasping operations. We are studying control structures for vision, tactile, and force feedback [9], and have demonstrated feasibility of adaptive control strategies for visual servoing. This work on sensor-based control is currently being extended to employ learning algorithms at the level of the motion primitive in order to improve performance by local adaptation in the face of uncertainty in the task environment. We have formulated an approach to quantitative description of task uncertainties using entropy methods [10], and have investigated the use of this *parts entropy* approach for planning strategies. We have also developed and demonstrated a new approach to arm signature analysis which improves the identification of kinematic models of manipulator structures and increases the resulting positioning accuracy [11].

Implementation of robotic systems in either a telerobotic or autonomous mode will require many of these planning, control, and manipulation capabilities. Task decomposition and control hierarchy have not been studied sufficiently for the telerobotic case. Development of motion primitives and planning of fine-motion strategies are important topics for research. The addition of adaptive and learning strategies to teleoperator systems is also important. The evolution of autonomous systems from telerobotic systems will require more effective models of human task planning strategies and task representation. The design of the components and tools of the space-based environment will depend on a consistent task representation which evolves to accept autonomous manipulation.

3. Assembly Task Representation

In our approach to assembly system design, [2,3,4], the planning of assembly of one product made up of several parts is viewed as a path search in the state space of all possible configurations of that set of parts. A syntax for the representation of assemblies has been developed based on *contact* and *attachment* relations. A decomposable production system implements the backward search for feasible assembly sequences based on a hierarchy of preconditions: (1) Release of attachments, (2) Stability of subassemblies, (3) Separability of subassemblies, including (a) Local analysis of incremental motion, and (b) Global analysis of feasible trajectories. Because there are many configurations that can be made from the same set of parts, the branching factor from the initial state to the goal state is greater than the branching factor from the goal state to the initial state. The backward search is

therefore more efficient and corresponds in this case to the problem of *disassembling* the product using reversible operations. The resulting set of feasible assembly sequences is represented as an AND/OR graph and used as the basis for enumeration of solution trees satisfying system and performance requirements.

Figure 1 shows an example of an AND/OR graph representation of assembly sequences for a simple product with four parts. Each node in the graph corresponds to a subassembly and is described in the representation by a relational structure using the syntax of contacts and attachments. The hyperarcs correspond to the disassembly operations, and the successor nodes to which each hyperarc points correspond to the resulting subassemblies produced by the disassembly operation. For most products, the assembly operations usually mate two subassemblies, and the resulting hyperarcs are typically 2-connectors as in this example.

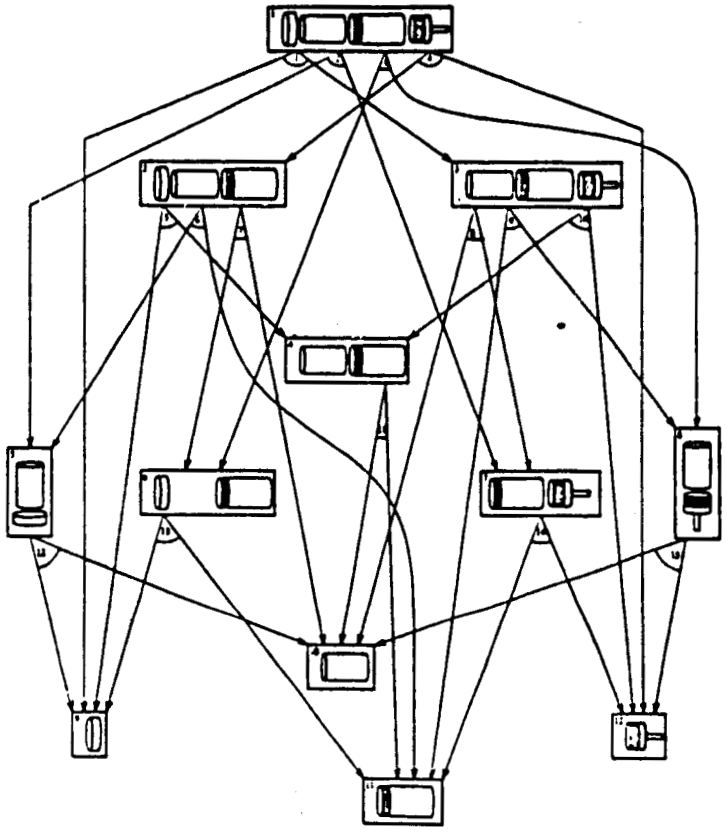

Figure 1. AND/OR graph representation of assembly plans for a simple product.

A solution tree from a node N in an AND/OR graph is a subgraph that may be defined recursively as a subset of branching hyperarcs from the original graph. The AND/OR graph representation therefore encompasses all possible partial orderings of assembly operations. Moveover, each partial order corresponds to a solution tree from the node corresponding to the final (assembled) product. The AND/OR graph representation therefore permits one to explore the space of all possible plans for assembly or disassembly of the product. The problem of selecting the best assembly plan may therefore be viewed as a search problem in the AND/OR graph space, and for some given evaluation function on the graph, generic search algorithms such as AO* [12] may be used. In practice, the development of such an evaluation function is very difficult since it would often depend explicitly on implementation issues such as choice of devices and underlying control strategies. We have explored the assignment of weights to hyperarcs using criteria of (a) operation complexity, and (b) subassembly degrees of freedom, or parts entropy [10]. Such an approach is viewed as a preliminary search procedure which may narrow the search space for later detailed examination using implementation details. In the simple examples studied, the resulting ranking of candidate assembly sequences was consistent with intuitive assessment of complexity.

The representation of assembly plans is particularly important for systems which do online planning or scheduling. Previous studies of online planning problems [13] have used discrete sequence representation or precedence diagrams of operations. In the precedence diagram formalism, typically no single partial order can encompass every possible assembly sequence. The AND/OR graph represents all possible partial orderings of operations, and each partial order corresponds to a solution tree from the node corresponding to the final product. We have illustrated the use of the AND/OR graph for online scheduling of a simple robotic workstation with random presentation of parts [2]. The resulting analysis showed a relative improvement in efficiency (number of operations required) from fixed sequence operation of 6% for precedence diagrams and 18% for the AND/OR graph. The principal advantage in this example was the reduced need for buffering and corresponding retrieval of parts.

The AND/OR graph representation provides a framework for the planning and scheduling of operations sequences. The problems of testing, disassembly, repair, and assembly all benefit from a unified representation which encompasses partial ordering of procedures. Preliminary search of the task space may reduce the candidate subtrees substantially, but the development of final plans typically involves directly the implementation and specification of the underlying devices and motions. In the next section we describe a tool for discrete task planning which facilitates exploration of alternative sequences of operations at the level of parts configurations.

4. Discrete Task Planning

A sequence of assembly or disassembly subtasks is implemented by performing operations on the parts using system resources such as robot hands, fixtures or sensors. The allocation of these resources and the synthesis of control programs to coordinate them must be developed in a second level of planning. In general, such operations require detailed motion planning of individual devices and is extremely difficult. In this section, we describe a definition of discrete operations which lend themselves to planning through manipulation of the configuration map relating input and output configuration states.

Any subtree of the AND/OR graph may be thought of as a *subtask precedence graph*,

and each branch of the subtask precedence graph defines a process in the configuration space of the parts. An assembly operation can then be defined by:

Assembly Operation:

Given $c^o = (C_i^o, C_j^o) \in C$,
control manipulation, sensing, and computation
to achieve $c^f = (C_i^f, C_j^f) \in T$, then
execute operation,

where T = tolerance set,

$T \subseteq C = C_i \times C_j$ for entities i, j,

is the set of configurations (region of configuration space [14]) for which an operation on i,j can be successfully performed.

This definition emphasizes the basic problem in assembly as the control over configuration uncertainty in order to meet tolerance requirements of successive operations. While it is possible to define probability distributions over configurations of parts, in practice, it is very difficult to accurately estimate such distributions, and it is cumbersome to propagate the effect of such distributions through successive operations in a sequence. The configuration map used here provides a tool to compute the effect of operations on bounding sets of configuration points.

A *bounding set* B(v) is defined as

B(v) = {possible outcomes of v}

where v is a bounded variable. We can define in turn:

Joint bounding set: $B(v_1, v_2, \ldots, v_n)$

Conditional bounding set: $B(v_1|v_2=\eta) = \{v_1|(v_1,\eta) \in B(v_1,v_2)\}$

Sum of bounding sets: $A + B = \{v|v = a+b \text{ for } a \in A, b \in B\}$

Scalar multiplication: $cA = \{v|v = ca \text{ for } a \in A\}$.

An operation which alters the configuration of a part may be described by a mapping between the initial configuration, Θ_i, and the final configuration Θ_f. An operation with a unique mapping occupies a single point in (C-space x C-space) and completely defines the change in configuration state of the system. In this case, planning of operations reduces to planning of unique trajectories in configuration space. As discussed above, such unique mappings are often of limited use due to the uncertaint y in configurations and the finite tolerance of operations. Then, states of the objects may be described by bounding sets of points in the configuration space.

The *configuration map* $M(A_i, B_j)$ describes a single operation which maps a bounded set of input points to a bounded set of output points:

$$M(S_1, S_2): \{S_1\} \rightarrow \{S_2\}.$$

The configuration map takes on logical values in (C-space x C-space) where each logical '1' defines a feasible mapping. The configuration map for a rigid part is a function of twelve dimensions, although in many cases these degrees of freedom are not of equal interest.

The usefulness of the configuration map representation of operations lies in the ease of combining sequential operations. An operation $M_1(\Theta_i, \alpha)$ followed by an operation $M_2(\alpha, \Theta_f)$ is defined as:

$$M_{12}(\theta_i, \theta_f) = M_1 M_2 = \cup_\alpha \{M_2(\alpha, \theta_f) \cap M_1(\theta_i, \alpha)\}.$$

Sequences of alternative operations may therefore be compared using simple relations.

The configuration map is particularly useful in cases where inputs and outputs may be partitioned into bounded sets. If we identify N subintervals B of the output space and N subintervals of A of the input space, then a symbolic mapping:

$$M' = \cup_j \{A_j \times B_j \mid M(\theta_i, \alpha) > 0\}.$$

defines bounded regions of the configuration map associated with transformations of bounded sets due to a given operation. A useful instance of the bounded set map occurs when we let:

$$A_j = \cup_{\alpha \in B_j} \{\theta_i \mid M(\theta_i, \alpha) > 0\}.$$

Then the configuration map

$$M' = \cup_j A_j \times B_j$$

is *rectangular* and the operation is completely defined by the symbolic map and the definition of the underlying sets.

The product of rectangular configuration maps is completely defined by bounding set operations:

$$M_1 M_2 = \cup_j {}^2B_j \times \{\cup_{k \in {}^{21}C_j} {}^1A_k\}$$

where

$${}^{21}C_j = \{k \mid {}^2A_j \cap {}^1B_k \neq \emptyset\}.$$

is the resulting configuration map product.

Figure 2 shows an example of a peg insertion operation in two dimensions, This type of problem has been studied from the point of view of trajectory planning in configuration space [15]. The configuration map shown in figure 2 is derived from such a trajectory analysis and summarizes the input-output relations in a manner which permits the resulting discrete operation to be integrated into task plan. A

different configuration map is developed for each set of discrete operations parameters, and the ability to form configuration map products permits search over the space of operations sequences. In figure 2, the x position of the peg is regarded as the independent variable of the map, and the initial z-position of the peg is fixed for a given configuration map. The operation moves the peg in a -z direction using a compliant move and directional uncertainty represented by the velocity cone [16].

The resulting configuration map in figure 2 has three output bands corresponding to successful insertion, miss-to-the-left, and miss-to-the- right. These three bands occur consistently for different parameter values. Five input bands may then be reconstructed and labelled defining a partitioning of the input configuration space. The resulting map may be 'rectangularized' as shown by the dotted areas, and in that form the symbolic mapping provide a complete description of the operation and a basis for search procedures.

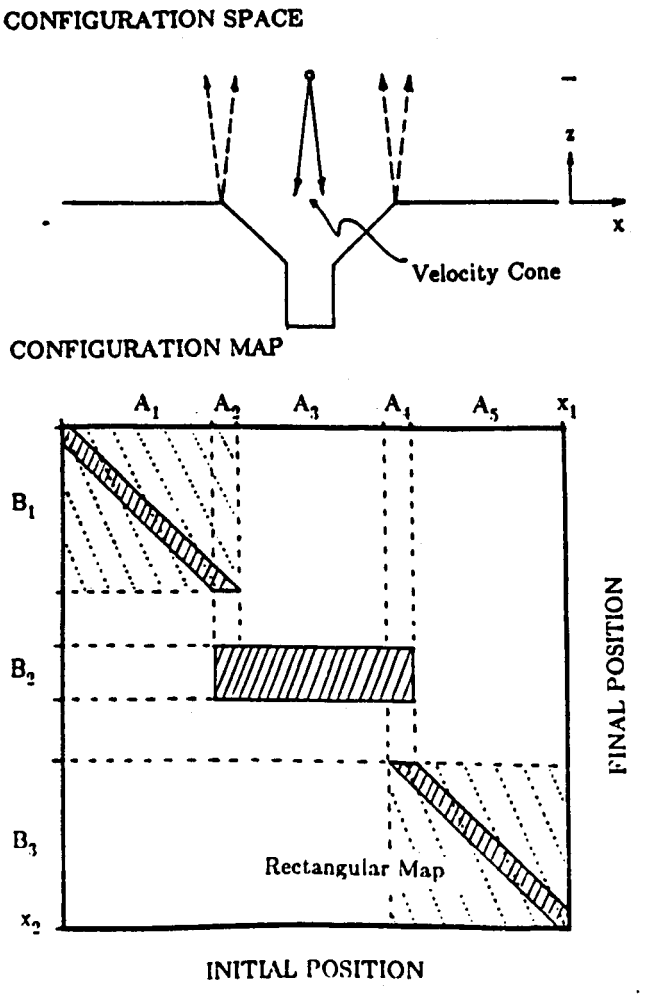

Figure 2. Configuration map for peg in hole example.

An example of a product of configuration maps is shown for a different set of operations in figure 3. Each of these maps is derived from our analysis of sliding objects [5,6,7] and corresponds to the orientations of a polygonal object being pushed by a two-dimensional fence of finite length. Equivalently, the object may be moving on a conveyor belt past a fixed fence. The independent variable in each map is the object orientation while the operation parameter is the fence angle. The uncertainty represented by the finite width bands in the maps is a result of the unknown support distributions of the objects. In [5,6,7] we derived bounds on the rates of rotation of such objects and have used these to compute the configuration maps for this example. The product of configuration maps therefore defines the bounds on the sets of orientations resulting from successive fence pushing operations, and can be used as a planning tool for designing sequences of fence push operations to achieve required goals.

For discrete tasks, the space of all operations sequences may be represented by a tree. Arcs correspond to operations, and each node represents a set of possible configuration states after execution of all the operations on the path from the root to that node. Figure 4 illustrates one such tree which corresponds to sequences of fence pushing operations for fences of different angles operating on the object shown in figure 3. The possible configurations of a part at a given node are obtained by multiplying the configuration maps for the operations on the path from the root to the node. Traversing the tree in order to search it is facilitated by the ease with which products of multiple configuration maps can be computed using the code sets.

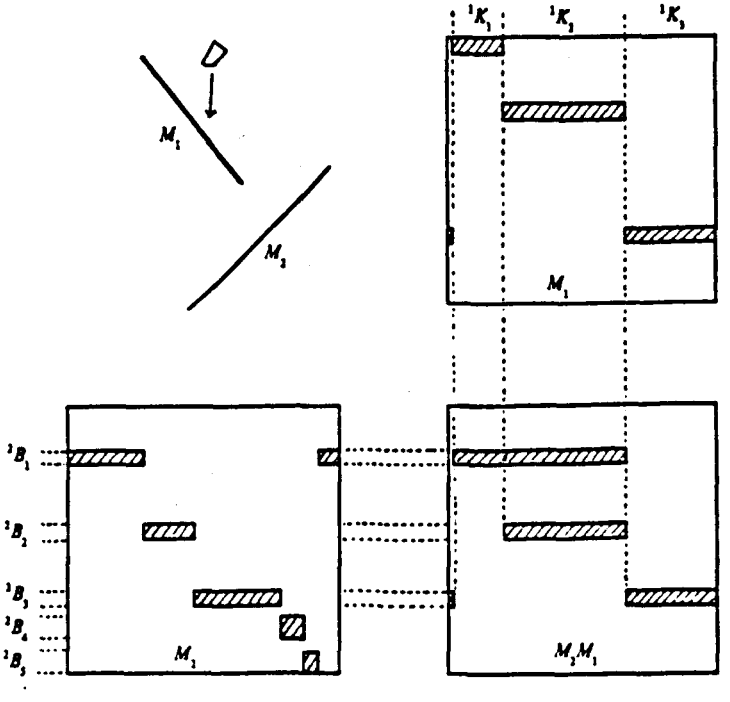

Figure 3. Product of two configuration maps.

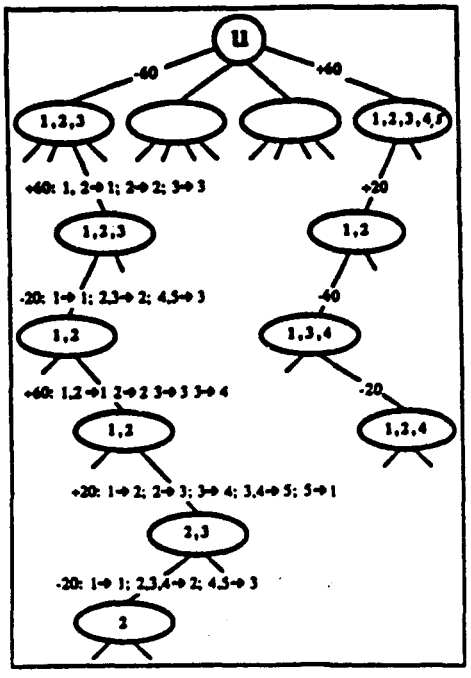

Figure 4. Tree search for operations sequence.

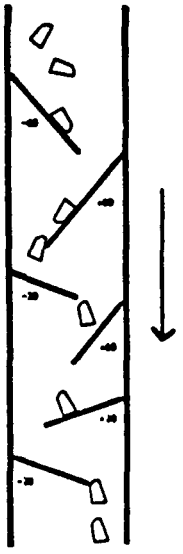

Figure 5. Resulting sequence of fence push operations.

Each node is labelled with the subset of the indices j of B for the bands B for the fence angle of the preceding arc. The goal of this task was to reduce the set of possible configurations to a narrow range of orientation, and a search strategy was implemented to reduce the number of output bands to one using the minimum number of operations.

Searching this tree of discrete operations exhaustively is computationally difficult due to the high branching factor which results from the available set of fence angles at each step. Two techniques have been developed to make this search feasible. First, there are systematic relations among bands for different operations parameters. Since there are only a few distinct code sets for the output arcs, it is often possible to systematically choose the subset of arcs which need to be followed among these outputs. Second, branches of the tree which develop code sets which have occurred previously in a shorter route may be pruned during search.

Implementation of these search techniques permits solution of the fence sequence design problem with the resulting design shown in figure 5. This parts feeder design will align parts with the geometry shown in figure 3 independent of the input orientation. Bounds on the orientation of the resulting single band are also derived from the procedure. The output part is then aligned for acquisition or handling by a robot. Computation for this search problem requires a few seconds of computation.

5. Conclusions

In this paper, we have reviewed several results in assembly representation, discrete task planning, and their relation to underlying control strategies. These methods of planning and manipulation are important for applications which will require autonomous systems to carry out complex tasks in diagnosis, repair, and assembly in space. The development of such analytical tools and their demonstration in prototype systems will be an important part of the evolution of telerobotic and autonomous systems for space applications.

REFERENCES

[1] A. C. Sanderson and G. Perry, "Sensor-based Robotic Assembly Systems: Research and Applications in Electronics Manufacturing," *Proceedings of the IEEE*, Special Issue on Robotics, Vol. 71, No. 7, July, 1983.

[2] L. S. Homem-de-Mello and A. C. Sanderson, "AND/OR Graph Representation of Assembly Plans," *Proc. 1986 AAAI Conference on Artificial Intelligence*, August, 1986, pp 1113-1119.

[3] A. C. Sanderson and L. S. Homem-de-Mello, "Task Planning and Control Synthesis for Flexible Assembly Systems," *Proc. NATO Int. Advanced Research Workshop on Machine Intelligence and Knowledge Engineering for Robotic Applications*, May, 1986.

[4] B. H. Krogh and A. C. Sanderson, "Modeling and Control of Assembly Tasks and Systems," *CMU Robotics Institute Technical Report*, CMU-RI-TR-86-1,] 1986.

[5] M. A. Peshkin and A. C. Sanderson, "The Motion of a Pushed, Sliding Object, Part 1: Sliding Friction," *CMU Robotics Institute Technical Report*, CMU-RI-TR-85-18, 1985.

[6] M. A. Peshkin and A. C. Sanderson, "The Motion of a Pushed, Sliding Object, Part 2: Contact Friction," *CMU Robotics Institute Technical Report*, CMU-RI-TR-86-7, 1986.

[7] M. A. Peshkin and A. C. Sanderson, "Robotic Manipulation of a Sliding Object," *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, April, 1986,pp 233-239.

[8] M. A. Peshkin and A. C. Sanderson, "Planning Sensorless Robot Manipulation of Sliding Objects," *Proc. 1986 AAAI Conference on Artificial Intelligence.*, August, 1986, pp. 1107-1112.

[9] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic Sensor-based Control of Robots with Visual Feedback," *IEEE Journal of Robotics and Automation*, in press, 1986.

[10] A. C. Sanderson, "Parts Entropy Methods for Robotic Assembly System Design," *Proc. IEEE Int. Conf. on Robotics and Automation*, March, 1984, pp. 600-608.

[11] H. W. Stone, A. C. Sanderson, and C. P. Neuman, "Arm Signature Identification," *Proc. IEEE Int. Conf. on Robotics and Automation*, April, 1986, pp. 41-48.

[12] N. J. Nilsson, *Principles of Artificial Intelligence*. Springer-Verlag, 1980.

[13] B. R. Fox and K. G. Kempf, "Opportunistic Scheduling for Robotics Assembly." 1985 IEEE International Conference on Robotics and Automation, pp. 880-889, 1985.

[14] T. Lozano-Perez, "Spatial Planning: A Configuration Space Approach." IEEE Transactions on Computers C-32, 2 (February 1983), 108-120.

[15] T. Lozano-Perez, M. T. Mason, and R. H. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots" Int. Journal of Robotics Research 3,1 (Spring, 1984), 3-24.

[16] M. Erdmann, "Using Backprojections for Fine Motion Planning with Uncertainty" Int. Journal of Robotics Research 5,1 (Spring, 1986), 19-45.